Lane Detection from Video Clips Using Binarization and Sliding Window

1st Fan Qingyuan
School of microelectronic
Southern university of Science and Technology
Shenzhen, China
Student ID 11812418
Email: fanqy2018@mail.sustech.edu.cn

2nd YUAN Tong
School of microelectronic
Southern university of Science and Technology
Shenzhen, China
Student ID 11810818
Email: yuant2018@mail.sustech.edu.cn

***Abstract*—As self-driving car become an emerging booming technology, several main technical issues began to be widely researched including obstacle detection, routine planning, road lane detection and danger estimation. Within all the issues, an important one is road lane detection, in other words, to extract the line on the road, so the computer can decide the exact direction which the car will go. In this project, a novel lane detection algorithm with binarization and sliding window technology have been proposed and achieved satisfactory result.**

***Index Terms*—Lane Detection, EE326**

Fig. 1. Result of lane detection using Hough transform

I. Introduction

The optical image based lane detection system is the key infrastructure of the driving assisting system, and also be regarded as the main component of the autonomous driving system. Specifically, lane detection provides reference information to the control of a car. In this article, we propose a method to detect and mark lane lines using a Binarization and Sliding Window technology.

II. Review

The detection of lanes remains challenging for several reasons: First the appearance of a lane is quite simple as a typical lane can be regard as a polynomial or curve with white or yellow colour filled. Such simplicity provides chance of false positive detection. Secondly, solid, broken, splitting or merging road markings are very common on highways or ordinary roads, which brings higher requirements for the robustness of the algorithm. Furthermore, contrast between lane marking and road surface will also reduces in raining, fogging or night scenario.

Several method have been proposed including traditional methods, which use the conventional computer vision and digital image processing algorithm like edge detection or Hough transform[1][2][5][8]; methods based on neural networks like LaneNet[9] and other NN based networks[6][3][4][7]. However, most of the algorithm has its own shortcomings: the method based on edge detection could not exclude the edge caused by the terrain on both sides of the road; method based on Hough transform could not solve the lane lines in curves, which is very common in the overpass section of the highway and urban way; method with NN and deep learning may have relatively better accuracy, but as this approach is still a form of pattern recognition, the network sometimes still recognize the curve like the lampshade as the lane line. In addition, since neural network-based models need to extract multiple parameters in the image, they often need to use a lot of special hardware such as GPU and CPU to achieve the available recognition speed, which is relatively costly.

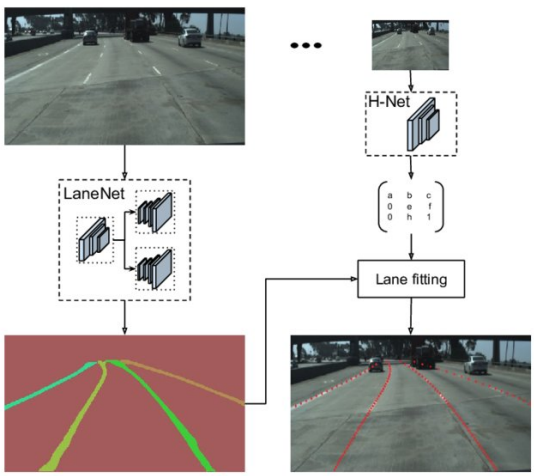

Fig. 2. Result of lane detection using NN (LaneNet)

A. Dataset

We use the CVPR2017 TuSimple dataset as the training and evaluating dataset for our model. The dataset of TuSimple Competitions for CVPR2017 contains 7000 1-second-long daytime highway video clips of 20 frames each, with the lanes on the final frame labeled.

III. Algorithm

The lane detection system we designed can be split into four parts: The perspective transform, The edge extraction and the sliding window recognition and the post-processing procedure.

Since python has many library functions for array computation and image processing, we choose python with cv2 and numpy library to implement our algorithm by programming code.

A. Perspective Transform

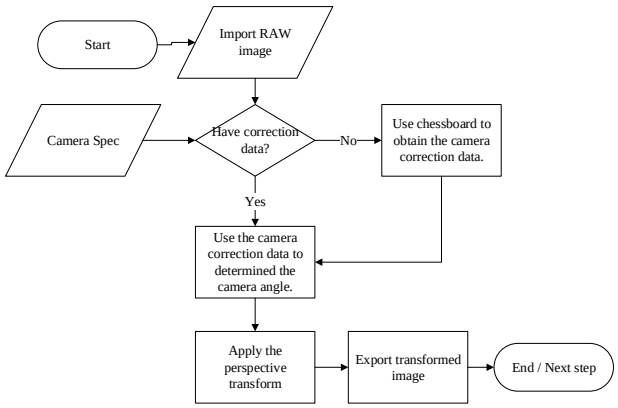

Fig. 3. Flowchart of perspective transform procedure.

1) Import the RAW image: Firstly, we extract the single frame from the video stream of the front camera of the car. In the training and testing stage, we just split the video clip obtained from the dataset and use the image file as the input of the whole function.

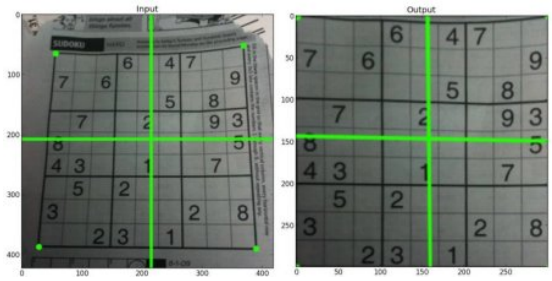

Fig. 4. Concept of perspective transform

2) Perspective Transform: From figure 3, we can see that the lower part of the ground in the camera picture can be treated as a plane, which we will only focus on this plane in the further step. Besides, preserve the vegetation and sky in the upper part of the image, and possibly interfere with the subsequent edge detection and segmentation.

To convert and correct the plane, the projection algorithm has been adapted to the input image. The general form of the perspective transform matrix can be written as:

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right) \tag{1}$$

while the θ_x, θ_y and θ_z is the value of Roll, Yaw and Pitch angle of the camera relative to the ground. The image passed the perspective transform is shown in Figure 3.

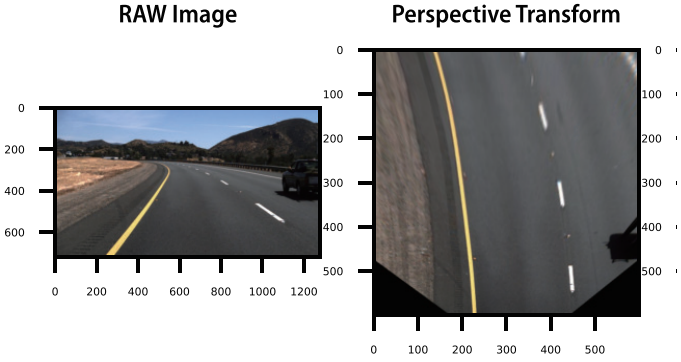

Fig. 5. Perspective transformation of the input RAW image.

B. Edge extraction and Lane Binarization

After the perspective transform, we need to convert an input image with three channels of RGB into a grayscale image containing only the lane lines, the general workflow of the edge detection and binarization is shown in the figure below:

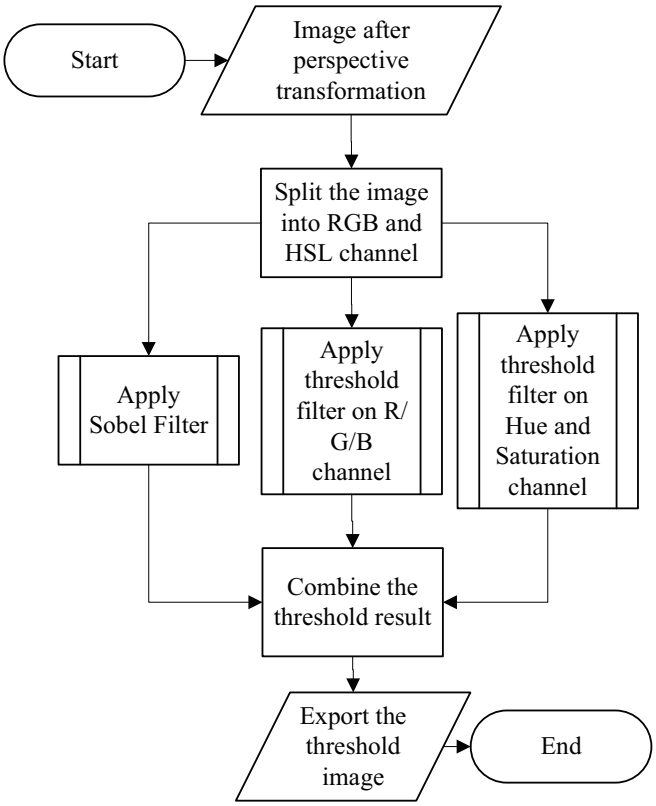

Fig. 6. Flowchart of Edge extraction and Lane Binarization.

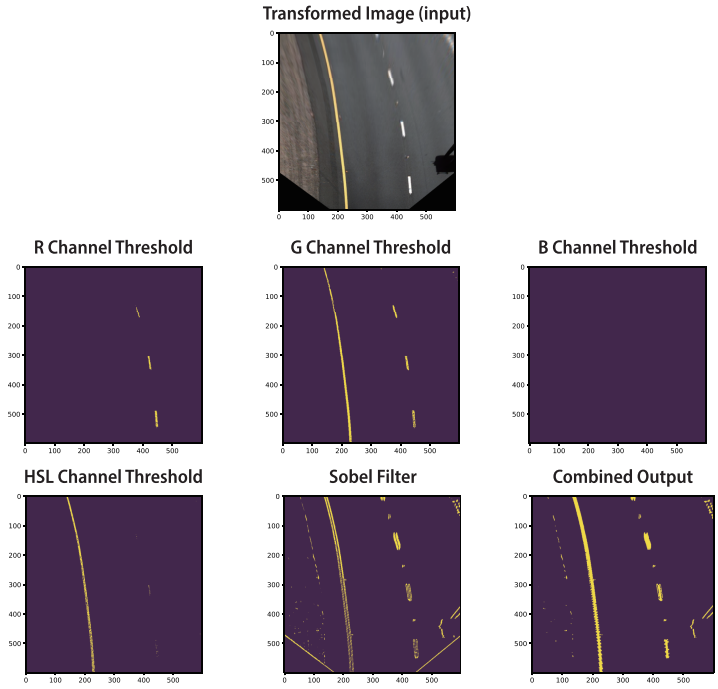

Fig. 7. Intermediate results and the combined result for Edge extraction and Lane Binarization process.

The transformed image will firstly passed an RGB filter to extract the ”bright” part of the image, while both yellow line and white line have a high brightness on the road. In the filter, image will be splitted into three colour channel, then apply the threshold filter, the threshold filter can be describe in equation 2, while T_l and T_h is the lower threshold and high threshold

$$y_i = \begin{cases} 1 & T_l < x_i < T_h \\ 0 & else. \end{cases} \quad (2)$$

The threshold result for each channel has been plotted in Fig 7. The combined output of the image is the sum of the input image.

C. Use Sliding Window to Detect the Edge

The most important part of the whole lane recognition system is use sliding window to extract the lane from the binarized image. As the flowchart in Fig 8, the input of the function is the image array after binarization, the output of the function is the axis array of the lane on the perspective transformed image.

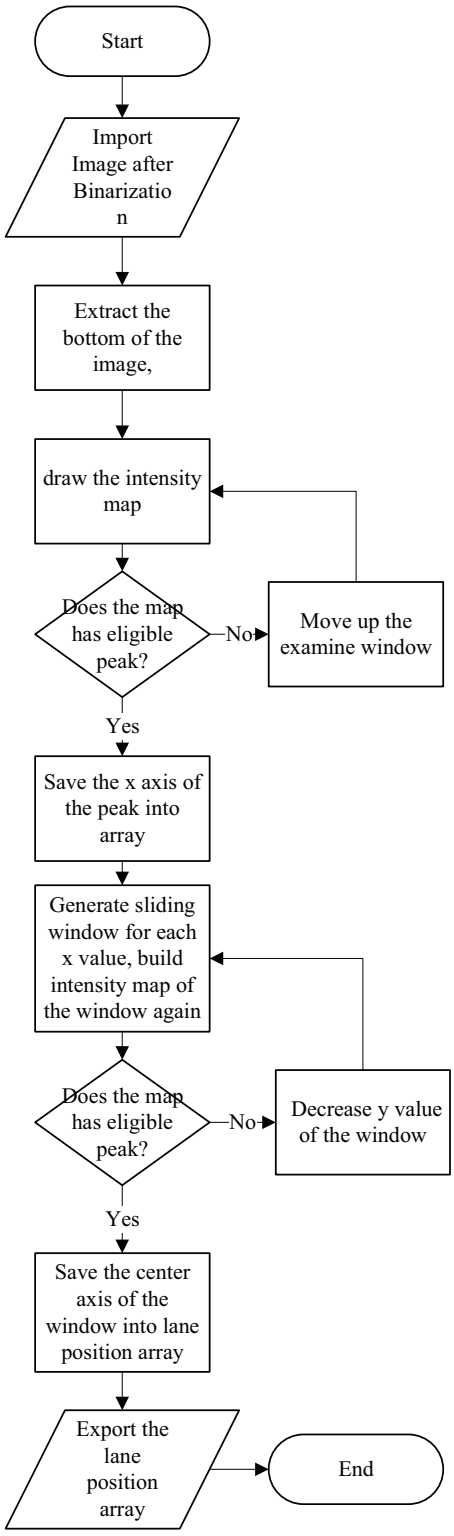

Fig. 8. Flowchart of the sliding window process.

1) Sliding Window: The image pass the binarization filter has the size of 600 px * 600 px. We firstly extract a slice of 200 pixel height and 600 pixel width from the bottom of the image, then we draw the histogram of image intensity versus x-axis. As the intensity of the lane line have been set with the intensity of 1 and other part have been set with the intensity of 0, the peak of the intensity map could be the most possible place that the lane line appear.

Therefore, to prevent the situation which the dash lane line not starting from the bottom of the image, if the program found the intensity of the peak does not meet the threshold we set, it will extract a new slice from the image with lower x axis (The maximum y-axis is at the bottom of the image).

After finding the x axis of the start point of the lane line, we cut a 20*50 image from the beginning of each lane line, the intensity map versus x-axis is adapted here again: If the function find the eligible intensity peak in the window, we append the current axis of the center of the window into the "lane point array", then decrease the y axis of the window, do the previous loop until the value of y axis reaches 0.

If the function find the value of the peak value is not eligible, e.g., the intensity is too low that such intensity could be caused by the noise, we just skip this window and decrease the value y axis and restart the loop above again until we find an eligible peak.

The left part of Figure 9 shows the point after the applying the sliding window algorithm to the image.

2) Regression Lane Curve: After extract the point set from the image, we can regression the lane line from the point and get the expression of the lane line for further processing. From our investigation, we use conic curve to fit the lane lines as most of the lane lines are part of a circle or a straight line. The result of the curve fitting can be viewing from the right part of Figure 9.

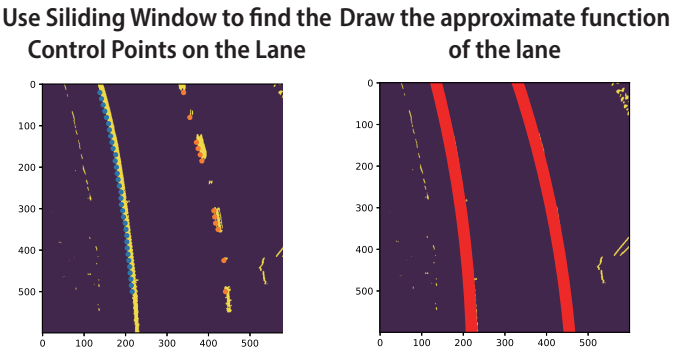

Fig. 9. The plot of the point found by sliding window algorithm and the road curve regressed by the point set.

D. (Optional) Plot the Regression Line Back

As the point set of lane lines can also be used as driver-assisted driving information viewed by the driver. We may also need to inverse mapping the coordinate of point set back to the coordinate system that the ground in the input image.

To implement the inverse transform, we only needs to find the inverse matrix of the perspective transform matrix mentioned in the earlier section. Once we have the transformed

lane coordinates and the approximation function, we can superimpose the lane markers on the original frame via OpenCV.

Fig. 10. The example recognition result of the input image.

IV. Result

According to our validation using the CVPR2017 dataset, our model have reached an overall accuracy of 84.6%, the model can be even more than 95% accurate on a straight path with good lighting conditions.

As previously stated, after find the start point of the lane line, the sliding window extraction process for each line can be done in parallel. With this optimization, we reached a recognition rate of 36 fps in the resolution of 1280 * 720 if we don't apply the optional inverse perspective transform at the last step.

Fig. 11. Some frames in the recognition result

V. Analysis & Discussion

A. Speeds up the model

From the model, we could find many processes that do not interfere with each other, which provide us the chance to speed up the execution time of the program by convert the serial process mentioned above into parallel. For instance, the function of apply different filter onto the perspective transformed image and the function to apply sliding window algorithm to each lane line can be executed in parallel and combine the result of sub-process at the end of the parallel execution process.

The python package numba provide us a way to parallel the algorithm quickly, which also bring us the ability about JIT (Just-in-time compilation) to compile some numpy operation before calling them. After the optimization, the frame-per-second (fps) of the model have been increased from 12 to 36 without causing any degree of sacrifice in the accuracy of the model.

B. Error Analysis

Although our model has reached a roughly usable level, there is still some corner case that our model could not handle. The first case is the scene that has many bright engineering facilities beside the road like the case in Figure 12. In this image, the Ice-cream cone have been exacted from the image together with the lane line, which caused the sliding window mistook the intensity information of the ice cream cone for the intensity information of the lane line.

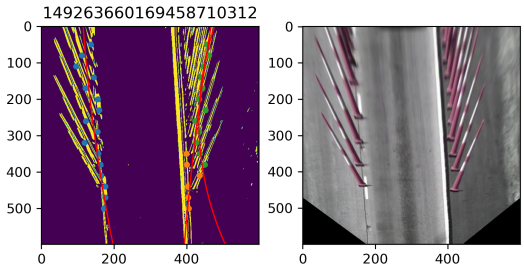

Fig. 12. Corner case: engineering facilities

Another issues of our model have been occurred while recognising the scene with shadow caused by the neighbour car cover which is illustrated in Figure 13. While the shadow of the car still has high contrast, when the shadow of the vehicle just covers the lane line, the sliding window could treat the edge of the shadow as the edge of the lane line, which could also leads to the offset lane coordinate set.

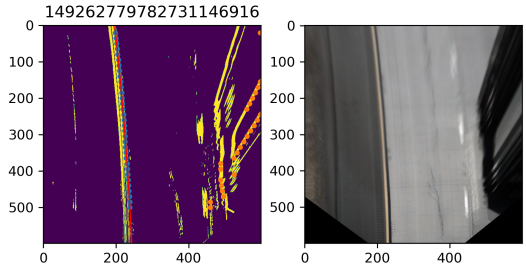

Fig. 13. Corner case: shadow across the lane

C. Future Work

Compared to binarization in color space, the multi-layer convolution layer may have a better performance on the case of high complexity. Such algorithm might be apply on the our model, which could use the NNs to remove the high frequency noise and the shadow before the sliding window processing to improve the model accuracy and robustness.

VI. Conclusion

In this paper, we proposed a lane detection model using Binarization and Sliding Window algorithm. The model first apply a perspective transform onto the raw input frame, then apply threshold on different channel of the image to get the binarization image array. Finally, the sliding window algorithm was adapted on the binarized image to extract the lane coordinate set. We also proved that the tradition computer vision method could still reach a usable result without the Graphics processing unit of Tensor processing unit. The future work will focus on adapting the benefits of neural networks and deep learning for detecting complex environments into the first two step of the model, which could further improve our recognition accuracy.

VII. Acknowledgements

We wish to sincerely express gratitude to:

- TuSimple for providing us with dataset.
- Prof. Yajun Yu for teaching us the fundamental concept of Digital Image Processing.
- The classmates that made suggestions to our algorithm on the opening report presentation.

VIII. Supplementary Information

A. Retrive the Code

The source code of this lab can be retrieved from https://github.com/sparkcyf/SUSTech-EE326-Digital-Image-Processing-Project (Github) or https://mirrors.sustech.edu.cn/git/fanqy2018/SUSTech-EE326-Digital-image-Processing-Project (Campus Git)

References

[1] Mohamed Aly. "Real time detection of lane markers in urban streets". In: *2008 IEEE Intelligent Vehicles Symposium*. IEEE. 2008, pp. 7–12.

[2] Hendrik Deusch et al. "A random finite set approach to multiple lane detection". In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2012, pp. 270–275.

[3] Alexandru Gurghian et al. "Deeplanes: End-to-end lane position estimation using deep neural networksa". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, pp. 38–45.

[4] Brody Huval et al. "An empirical evaluation of deep learning on highway driving". In: *arXiv preprint arXiv:1504.01716* (2015).

[5] Yan Jiang, Feng Gao, and Guoyan Xu. "Computer vision-based multiple-lane detection on straight road and in a curve". In: *2010 International Conference on Image Analysis and Signal Processing*. IEEE. 2010, pp. 114–117.

[6] Jihun Kim and Minho Lee. "Robust lane detection based on convolutional neural network and random sample consensus". In: *International conference on neural information processing*. Springer. 2014, pp. 454–461.

[7] Jun Li et al. "Deep neural network for structural prediction and lane detection in traffic scene". In: *IEEE transactions on neural networks and learning systems* 28.3 (2016), pp. 690–703.

[8] Marcos Nieto et al. "Robust multiple lane road modeling based on perspective analysis". In: *2008 15th IEEE International Conference on Image Processing*. IEEE. 2008, pp. 2396–2399.

[9] Ze Wang, Weiqiang Ren, and Qiang Qiu. "Lanenet: Real-time lane detection networks for autonomous driving". In: *arXiv preprint arXiv:1807.01726* (2018).